Resin Cookbook - Article: **Securing Resin**

Securing Resin can involve not only access within your web-app, but access to Resin itself. In this article we will look at the following ways to secure your Resin deployment:

1. IP address protection
2. Hiding URLs
3. HTTP Basic Authentication
4. Disabling Protocols

IP address protection

If you want to restrict a web app access, you can use an <resin:Allow> tag with a <resin:IfNetwork> tag to only allow my IP ranges. The configuration looks like the following:

```
<web-app xmlns="http://caucho.com/ns/resin"
         xmlns:resin="urn:java:com.caucho.resin">
  ...
  <resin:Allow url-pattern="/admin/*">
    <resin:IfNetwork value="192.168.17.0/24"/>
  </resin:Allow>
  ...
</web-app>
```

After matching the url-pattern, <resin:Allow> checks all the predicates it contains, like <resin:IfNetwork>, and if all match it allows the request. (Otherwise, processing goes into a default-fail mode.)

Resin's <web-app> configuration allows for multiple resin:Allow in use with any of the resin:ifxxx tests:

```
<resin:Allow url-pattern="/foo/bar/baz/*">
    <resin:IfNetwork value="10.0.0.1"/>
  </resin:Allow>

<resin:Allow url-pattern="/foo/bar/*">
   <resin:IfNetwork>
      <resin:value>10.0.0.1</resin:value>
      <resin:value>10.0.0.2</resin:value>
   </resin:IfNetwork>
 </resin:Allow>
```

Behind a loadbalancer that is handling SSL, Resin can assume all connections to be secure. If you have specific pages that you would like to enforce as https behind something like a loadbalancer, then you would issue redirects so that these pages have the https.

If you wanted to issue http -> https redirections to specific pages, you would add a setRequestSecure with a regexp that matches the encrypted login page:

```
<web-app xmlns="http://caucho.com/ns/resin"
         xmlns:rewrite="urn:java:com.caucho.rewrite">

    <rewrite:SetRequestSecure regexp="^/loginPage.html"/>

  </web-app>
```

Another approach to this, can be to set your <web-app> tag to secure as follows:

```
<web-app secure="true">
    ...
  </web-app>
```

/resin-admin
By default, access to /resin-admin is granted to the local network only:

```
 <resin:if test="${! resin_admin_external}">
     <resin:IfNetwork>
       <value>127.0.0.1</value>
       <value>192.168.0.0/16</value>
       <value>[::1]</value>
     </resin:IfNetwork>
```

If you are working with a clustered deployment, you can externalize the properties into resin.properties for added deployment convenience. Adding variables such as admin_network_range_0 and admin_network_range_1.

In that case, resin-web.xml will only need to be distributed when values are added or removed. Reference values in resin-web.xml with <value>${admin_network_range_0}</value>

Alternatively, you could add the following construct to resin-web.xml and accompany with an if-network.xml document:

```
<resin:Require>
   <resin:if test="${! resin_admin_external}">
     <resin:IfNetwork>
          <resin:import path="WEB-INF/if-network.xml" optional="false"/>
     </resin:IfNetwork>
   </resin:if>

   <resin:if test="${! resin_admin_insecure}">
     <resin:IfSecure/>
   </resin:if>
  </resin:Require>
```

the WEB-INF/if-network.xml's structure should look like the following:

```
<resin:IfNetwork xmlns="http://caucho.com/ns/resin"
                 xmlns:resin="urn:java:com.caucho.resin">
  <value>127.0.0.1</value>
  <value>192.168.0.0/16</value>
</resin:IfNetwork>
```

In this case you will only need to distribute modified resin-web.xml once and the IfNetwork change will go into if-network.xml and redistributed when changed.

Hiding URLs

Some URLs are meant to be hidden, like WEB-INF or META-INF or some configuration directories in PHP. You can use a <resin:Deny> tag to forbid access to any set of URLs.

```
<web-app xmlns="http://caucho.com/ns/resin"
         xmlns:resin="urn:java:com.caucho.resin">

  <Deny>
    <resin:url-pattern>/hidden/*</resin:url-pattern>
    <resin:url-pattern>*.hidden<resin:url-pattern>
  </Deny>

</web-app>
```

The two url-patterns illustrate that <resin:Allow> and <resin:Deny> can match multiple URLs at once. You can even add http-method to restrict the methods that you're protecting.

Resin's security processing follows a first-match algorithm. The first <resin:Allow> allows the request, or if <resin:Deny> matches first, it denies the request. If the url-pattern and http-method match, but the predicates fail, Resin goes into default-fail mode for <resin:Allow> or default-allow mode for <resin:Deny>. If no other rules match, Resin will take the default.

HTTP Basic Authentication

If you want a simple login authorization for a quick and dirty project, you can use HTTP basic authentication and an XmlAuthenticator. Resin's login is split into a Login class and an Authenticator. The Login is responsible for managing the HTTP security protocol: basic authentication, or password authentication, or multi-request protocols like digest. The Authenticator is responsible for checking passwords.

In this case, the example is using <resin:BasicLogin> in order to skip writing extra JSP or PHP pages, and the <resin:XmlAuthenticator> instead of querying a database.

```
<web-app xmlns="http://caucho.com/ns/resin"
         xmlns:resin="urn:java:com.caucho.resin">

  <resin:Allow url-pattern="/secure/*">
```

```
    <resin:IfRole name="*"/>
  </resin:Allow>

  <resin:BasicLogin/>

  <resin:XmlAuthenticator>
     <resin:user name="harry" password="uTOZTGaB6pooMDvqvl2Lbg==" group="user"/>
  </resin:XmlAuthenticator>

</web-app>
```

As with all the WebBeans-style configuration, com.caucho.security.XmlAuthenticator is the actual class you're configuring, so the JavaDoc itself can provide documentation:

http://javadoc4.caucho.com/com/caucho/security/package-summary.html

Disabling Protocols

Modifying OpenSSL to specify allowed cipher suites and protocols can be done in the Resin application server configuration file, resin.xml, in the <openssl> block.

```
<cluster id="web-tier">
<server id="...">

<http port="443">
  ...
  <openssl>
    <certificate-key-file>keys/your_domain.key</certificate-key-file>
    <certificate-file>keys/your_domain.crt</certificate-file>
    <certificate-chain-file>keys/chain.txt</certificate-chain-file>
    <password>test123</password>
    <cipher-suite>ALL:!aNULL:!ADH:!eNULL:!LOW:!EXP:RC4+RSA:+HIGH:+MEDIUM</cipher-
suite>
    <protocol>-all +sslv3 +tlsv1</protocol>
  </openssl>
</http>

</server>
```

With JSSE you would add the protocols supported to your <jsse-ssl> tag:

```
<jsse-ssl>
  ...
<protocol>TLSv1.2,TLSv1.1,TLSv1</protocol>

</jsse-ssl>
```